The case for open source software

The Case for Open Source Software: The Interactional Discourse Lab

Seongsook Choi

Moray House School of Education, University of Edinburgh
E-mail: s.choi@ed.ac.uk

Computational techniques and software applications for the quantitative content analysis of texts are now well established, and many qualitative data software applications enable the manipulation of input variables and the visu- alization of complex relations between them via interactive and informative graphical interfaces. Although advances in text analysis have helped researchers mine text data for semantic content and identify language patterns in text withgreater facility, interactional dynamics and patterns of talk have been neglected. This article introduces a new open-source tool, *Interactional Discourse Lab.* This tool is designed to map dynamics in spoken interaction and to represent them ineasily accessible visual form, capturing aspects such as the frequency and pat- terning of exchanges, and the distribution of turns and discourse features. It is designed to contribute, with other analytical tools such as those used in text analysis, to the development of *interactional topographies.* The paper sets the tool within a wider case for the development of open-source software in applied linguistics as a platform for methodological innovation.

INTRODUCTION

Both commercial and bespoke software tools for text analysis, content analysis, quantitative and qualitative analysis are widely available and widely used, and the multitude of functions they offer make them a convenient option [see Lewin and Silver 2007; Evers *et al.* 2011 for a detailed history of Computer-Assisted Qualitative Data AnalysiS (CAQDAS) and its functionalities, and http://caqdas.soc.surrey.ac.uk/ for details of available software tools].[1] While some researchers have expressed doubts about the value of such diversity, especially given the time and effort involved in user training (Silver and Lewins 2007, 2014), a more serious problem is that most have functions designed to serve one methodological paradigm, requiring users to work within the parameters determined by the producer. Moreover, the aggressivemarketing of software creates a situation in which institutions may be able to support only one package (Fielding and Lee 2002) and the cost of some soft- ware products may be prohibitive. For this reason, many researchers are forced to prioritize their choice of CAQDAS tools in terms of institutional availability rather than the precise relevance of its functionality to their work, problems

that may be exacerbated by a paucity of detailed documentation of analytic and technical procedures, and critiques of software utility (Silver and Patashnick 2011).

Although the advantages of the use of software for methodological convergence have been recognized by researchers in some fields (Fielding and Cisneros-Puebla 2009), the main response in the social sciences has been the development of ad hoc task-function or research-project-specific tools by the few who can programme or have the funds to employ a programmer. These offer a precisely targeted response, but this in turn limits the range of their applicability, which means that most researchers are forced to rely on proprietary alternatives. However, there is recent evidence of a more collaborative approach to developing software tools in the area of corpus research, a notable example being AntWebConc (Anthony *et al*. 2011), software built through collaborative efforts of teachers, researchers, and programmers which has led Anthony (2013) to argue the case for collaborative community effort in developing corpus tools. This builds on earlier arguments by Biber *et al.* (1998), Mason (2008), Gries (2009), and Weisser (2009) encouraging corpus linguists to learn programming languages tobuild tools for specific tasks that existing software cannot provide.Garretson (2008) offers information for English language researchers withprogramming skills on how to design software.

The present article builds on this pioneering work. It discusses the advantages, challenges, and issues associated in developing and using open-source software (henceforth OSS) for applied linguistic analysis more generally, arguing that this represents an opportunity for a radical re-orientation inthe use of computer-aided analysis, holding out the prospect of a new levelof methodological innovation based on collaboratively evolving platforms.In making the case for distributed code development, it situates research methodology within a broader theoretical shift in applied linguistics (AL) that approaches language not from the perspective of individual cognition but in terms of co-constructed social action. The article begins with an explanation of the nature of OSS and a consideration of its benefits, proposing a model for its use that maximizes its methodological potential. As an illustration of the model in action, the article then introduces a new open-source tool, the interactional Discourse Lab (IDLab), a program developed by the author using R, an open-source statistical programming language. The tool is designed to map dynamics in spoken interaction and to represent them in easily accessible visual form, capturing aspects such as the frequency and patterning of exchanges, and the distribution of turns and discourse features. It is designed to contribute, with other analytical tools such as those used in corpus analysis, to the development of *interactional topographies*: different landscapes of interaction or patterns of talk within specific domains andthe distinctive interactional contours that have been developed by groupsover time.

OSS IN ACADEMIC RESEARCH

Writing bespoke software is now an integral part of the research process in the physical sciences and although researchers have traditionally kept close control over their programs there is now an increasing requirement to publish their source code (i.e. the inner working of the program) in academic papers. At first sight, it may seem that releasing the source code would risk affecting authorship rights or ruining a potential commercial opportunity: someone can simply copy the software, rebrand it, and sell it under their own name. There are, however, many advantages to releasing OSS for the developer, for the users, and, in the long run, for science itself, which explains the change in publishing practices.

The use of OSS is well established in computer sciences and more widely in the physical sciences. Researchers in AL and the social sciences, on the other hand, seem to be largely unfamiliar with it, with the exception of a small number who have programming skills and are able to write a tool to execute tasks that are specific to their project needs and not otherwise achievable using free or commercialized packages. This section explains what OSS is and highlights its established features and practices in the physical sciences. The benefits of OSS compared with closed-source software and the potential contribution of the former to AL and social sciences research are also discussed.

What is OSS?

Software is written in abstract coding, programming languages (such as Java, R, or Python) in which the programmer needs to be fluent. The instructions written in this language are then turned into a list of commands that a computer can understand, through a process called compilation. A user of the software requires only the compiled version (or '*binaries*') to run it; it contains all the commands the computer needs to follow in order to execute the task. Binaries are designed to be run by a computer but they are very difficult for a human to inspect because the original set of instructions has been reduced to a long sequence of 0s and 1s.

A program is said to be open-source when its source code is made publicly available. Most commercial and research software tends to be distributed as closed-source: the users have access to only the binaries (the executable version) of the program, and while they are able to run the software, they cannot inspect or modify it. In contrast, open-source programs publish the original code that produced the software, known as the source code. This is written in a programming language that can be read and understood by those familiar with the relevant language, as well as modified.

Since the source code is publicly available, it is possible for interested and able users to contribute to the project, from reporting bugs and improving algorithms to extending its original version. Many tools and services are

now available that facilitate the contributions of people not involved in the original project or software development. Contributions can take the form of code improvement, reporting bugs, and requesting features. Examples of such tools and services include distributed version control software like Git and Mercurial that make it easy to submit requests and patches, and services such as Github for developers, as well as forums and mailing lists for users. All these are popular ways of quickly building a community of people inter- ested in the group development of the most efficient program—an arrange- ment that would be unthinkable in a closed-source scenario.

The case for open source

While these advantages of OSS make it an appealing option for applied linguists, it is the contention of this article that taken together the use of OSS has the potential to form the basis of a new developmental and methodological approach with its primary focus on decentralized, distributed development, flexible membership, and an on-going process of community-wide software testing and debugging in a process reminiscent of Cowley's (2007: 119) concept of 'distributed language' and the idea that '[m]ulti-agent models can be used to explore hypotheses about patterns that emerge from repeated interaction'. In what follows, these advantages are grouped under the acronym 'VARIES'.

Visibility

Wren (2008) has shown how rapidly dedicated academic software vanishesafter the research has been published. When such software is also published, itsaves researchers' time and effort involved in rewriting the same software.Publishing the code on a public repository and attracting users into forminga community makes research software more visible: the software is easier totrack down, the community itself advertising it through word of mouth andresearch practices. This allows more people to become involved in its develop-ment. When this is the case, the feedback and interest from the community can be quantified by the original developer and used as evidence of the impactof their work in the same way as citations have been used traditionally (e.g.number of contributors, number of downloads, changes). In fact, a growingnumber of publications are recognizing the effort spent on programming bysupporting the publication of the code that comes with a research manuscript(Morin *et al.* 2012a). For example, both *Biostatistics* (Peng 2009) and *PLOS Computational Biology* (Prlić and Lapp 2012) have made editorial decisions to encourage code publications.

Adaptability

The adaptability of OSS allows the involvement of new researchers with different needs, which in turn means that the software itself is less likely to fossilize

and become obsolete. For example, OSS developed in cloud computing is used in cancer research (https://blog.cloudflare.com/cloudflare-fights-cancer/). It also is not uncommon for the original developer to become less involved in their project, while the software is kept alive by the community it built. Examples can be found with Gnome (a Linux desktop environment) or many smaller projects (e.g. uBlock, a browser add-on).

Reactivity

Compared with an equivalent closed-source or proprietary software, an open-source alternative is also more reactive to user needs and the pace of research. With more people involved, improvements are likely to happen quickly, and thus users are more likely to give feedback as they see their issues addressed faster than if one single, probably overworked, developer were to do it all.

Inclusivity

Because they use the software in their own research, the contributors have a stake in making it work, encouraging what has been described as user-driven innovation (Hippel 2001; Lerner and Tirole 2002). An illustration of this is available in the form of the natural language processing task view (http://cran.r-project.org/web/views/NaturalLanguageProcessing.html), a manually curated collection of R packages dedicated to NLP.

Enhanceability

The quality of the code is likely to improve for two reasons: more people are involved, some possibly having greater programming experience than the original coder, and there is added pressure on developers to write correct code because they know that their work will be publicly scrutinized. Developers in the public spotlight are less tempted to cut corners and more likely to follow good software engineering principles (modularity, unit-testing). This improves the original code and makes its output more reliable (with fewer bugs and a clearer understanding of how the results are derived). Such practices can help researchers with no formal training in programming feel more sure about releasing their code public because of the natural peer review process that occurs when the code is released. Open-source code also can then profit from the synergy of multiple developers (Sornette *et al.* 2014), and the larger a community is, the more likely bugs will be spotted and fixed: '*given enough eye-balls, all bugs are shallow*' (Raymond 2001: 33).

Speed

An extra advantage of producing code of good quality is that it can be re-used for other open-source projects without re-inventing the wheel (Sojer and Henkel 2010), thus accelerating the pace of research. Additionally, while

monolithic proprietary software traditionally moves slowly, with at best a yearly release, this is in stark contrast with an open-source community like R in linguistics: the latest research tools and techniques are available for anyone to use, sometimes as soon as, or even before, they are officially published.

Reservations relating to OSS

Two frequent concerns about OSS in academic settings are the risk of being 'scooped' and the seemingly obvious forfeiture of future commercialization. Since the software code is in the open, it is indeed possible for someone to take it and either produce a scientific paper before the original author using this head-start, or even simply claim it as their own. While possible, this scenario rarely if ever plays out in reality. First, precisely because the original work is a matter of public record, it is easy to assert 'prior art' (the right to authorship). Secondly, there is overwhelming evidence that these are not treated as serious threats: one need only consider very popular free and public pre-print repositories like arXive (http://arxiv.org/) and biorXiv (http://www.biorxiv.org/). Thousands of manuscripts are submitted to these every week by academics eager to share their work with their peers in advance of official recognition through journal publication.

While open-source code can be published as is, it is best practice to accompany it with an OSS licence clarifying what other developers are allowed to do with the original code (Morin *et al.* 2012b). OSS licences do not necessarily prevent future commercialization; for example, one can decide to make the software free for research and education organizations but not for commercial entities. In fact, rather than selling the software itself, a company's business model can be based on services (e.g. training, maintenance, bespoke versions), and many successful companies use OSS as their core product [e.g. Red Hat (linux), Oracle (MySQL), and Revolution Analytics (R)].

Given the advantages of OSS, it may be surprising that most software, and commercial software in particular, are only available as closed-source. The case for the closed-source alternative draws on a number of potential disadvantages of OSS:

1. A greater emphasis is put on usability in commercial software; while OSS are typically started to 'scratch an itch' and focus on functionality, commercial companies invest a lot of effort in ensuring high-quality user experience and robustness to misuse (making it 'idiot-proof'). These aspects are often either not to be found in OSS and are in any case to replicate due to the lack of naïve users among the interested parties.
2. There is still a large degree of scepticism over software quality, the assumption being that something given away must have low value, or inversely, if it is expensive, it must be of high quality or it would not survive in the market place. The view of a commercial software employee

(although she did subsequently apologize) is not atypical: 'We have customers who build engines for aircraft. I am happy they are not using freeware when I get on a jet.' (http://www.nytimes.com/2009/01/07/technology/business-computing/07program.html). Given the prevalenceof this opinion, even an OSS advocate might think twice before publish- ing his/her code and risking alienating potential customers.

3. Some also consider that publishing code is a potential security risk: a malicious developer can identify a weakness in the software and exploit it to his/her own gain, as in data theft or sabotage (Payne 2002). Keeping software closed-source seems to remove or at least diminish this possibility, the so-called 'security through obscurity' assumption.

Towards more reproducible research

The current situation in AL and, more broadly, social science research typically consists of unpublished data and black-box software (Morin *et al.* 2012b; Ram 2013) and as Joppa *et al.* (2013) report, even in the physical sciences, '*software code is not formally peer-reviewed*'. There is a call for science to move towards the sharing of data (van Assen *et al.* 2014) for increased transparency and reduced publication bias, not least because exposing the process by which the reported results are arrived at allows evaluation of the published evidence (Peng 2011). More broadly, however, OSS offers the prospect of a reconfiguration of research activity in which software programmes are used.

As a number of AL researchers have begun to argue (Porte 2012), reproducible research is an essential step towards more reliable results and better science, and the transparency of OSS—the ability to reproduce a researcher's results given their data and their code—is crucial for this. Indeed, one could consider OSS as being continuously informally peer reviewed by interested reviewers, who are also the end users. As each new user adapts the source code, they thereby (i) test the original and (ii) expand the library of available packages. The need to adapt is driven by methodological challenges that are different from those of the original project, which means that innovative thinking may be required. If the link between research method and software development is also made transparent, the fund of methodological experience is also thereby enriched.

This allows an element of shared methodological development. In a traditional model, methodological decisions are typically made by those involved in a particular project and reported on completion of the project. However, the process of continuous evolution characteristic of OSS means that new updates can become available when methodological decisions are, as it were, 'in process' and can then be built into this and themselves inform further developments in terms of both research method and computational technique. This offers the possibility of a radical revision of methodological practice away from individual or team endeavour and towards a more distributed model based on

collaboratively evolving platforms. The first steps towards this have already been taken in the physical sciences, but the human sciences have yet to embrace it. The interdisciplinary orientation of AL and the breadth of disciplines on which it draws make it an ideal candidate to lead the way in exploiting the potential of this model, but in order for this to happen there needs to be a radical re-orientation of attitudes to what counts as ownership and originality.

Practical advice on developing OSS

In order to illustrate what this process might involve, brief extracts from research based on an OSS tool developed by the author will be presented, but this needs to be seen in the context of practical steps involved in OSS development. This section summarizes these and is intended as no more than a guide addressed to prospective developers.

1. There is no rule about when to publish developers' code. Some start as early as possible ('*open-development*', Prlić and Procter 2012) but opening the code can be done at the time of the publication or even later. This is to a large extent up to the developer and their collaborators.
2. The simplest way to publish code is to upload it to a webpage (e.g. the author's institutional webpage) and it should be bundled with an OSS licence. The range of choice can seem bewildering, but a developer's decision should be based on how they want their code to be used by future projects: from granting future users complete freedom over future projects (a *permissive* licence such as Berkeley Software Distribution or Massachusetts Institute of Technology), to allowing access to future uses of their code only if they are under specific licensing restrictions (a *copyleft* licence such as GNU (the acronym for 'GNU's Not Unix!') General Public License or Lesser GNU General Public License). Copyleft (a pun on 'copyright'; a copy of the license is left with the source code) licences might seem more restrictive, but they exist to ensure that derived projects stay in the realm of OSS.
3. The combination of multiple OSS with different licences is also another factor to consider; some licences are not compatible, and software with a permissive licence might be preferred over one with a more restrictive licence. Morin *et al.* (2012b) provide a detailed explanation of the range of available licences and the rationale for selection.
4. While posting the source code on your webpage is a good first step, there are better ways for publishing it that are also more amenable to building a community and receiving feedback: Version Control Systems (CVS) and bug tracking software. CVS allow developers to keep track of changes in the code by taking a snapshot at a point in time and annotating the change. Thus, it is very easy to go through the history of an algorithm and explain, for example, why a particular choice was made. The most popular CVS currently are git, mercurial, and subversion.

5. The history of all changes is posted to public repositories hosted on a server from which developers can download a copy. In the case of OSS, the repository is public and anyone can access it, as well as contribute to it. A number of companies offer free hosting for OSS, examples including GitHub (github.com), BitBucket (bitbucket.org), or sourceForge (source-forge.net). CVS are not only good software engineering practice (Ram 2013; Wilson *et al.* 2014), they also make it much easier for external developers to contribute to an open-source project. Typically, an inter- ested third-party downloads ('pulls') a copy of the source code, works on it, and submits the patch to the repository's owner, who can decide whether to include ('push') it in the current version.
6. CVS make the collaborative process of editing the code much easier than, say, doing it by email or a shared folder. Moreover, since every change is digitally signed by its creator, attribution is clear. Most repository hosts also include a bug tracking service, where people can report bugs or request features. Each post is triaged and addressed by the repository's owner or anyone else, as the case may be. This is ideal for involving non- technical users, who cannot contribute on the code engine itself. It can be also coupled with more traditional approaches like a forum or a dedicated mailing list (Prlic´ and Procter 2012).

INTERACTIONAL DISCOURSE LAB

The case for OSS has so far been made in general terms, but in order to illustrate how this resource might be used in the context of AL, this section provides an example of a newly developed OSS tool. It begins with a brief justification for the tool in the context of other programmes that are available, moves on to describe the tool itself, and concludes with an illustration of how its outputs have contributed to a research project focusing on interdisciplinary team meetings.

The IDLab (www.interactionaldiscourselab.net), a free open-source visualization tool, captures the interactional dynamics of talk-in-action using both qualitative and quantitative methods. It automatically generates interactive visualizations of the patterns of interactions from an input transcript that has been tagged by the user of the tool. The IDLab processes the tags to pro- duce visual representations of the information using R, a statistical program- ming language (http://r-project.org). The generated visuals are then displayedin three separate panels: speakers and tags, interactions, and timeline, each panel updating the relevant statistics and graphs according to the tags selected by the user.

Computational techniques in quantitative content analysis have been used since the 1950s (McEnery and Hardie 2011). Nowadays well-established software applications such as AntConc and WordSmith offer semi-automated text analysis techniques (e.g. concordance, collocation, and cluster analysis) that

speed up and simplify the text analysis process. These techniques are used to mine text data for semantic content and to identify language patterns in the text. CAQDAS applications (e.g. Atals.ti, MAXQDA, NVivo) arrived on the scene later and enhanced traditional analysis methods such as Roter's Interaction Analysis System (Roter and Larson 2002) and Bale's Interaction Process Analysis (Bales 1950) by allowing researchers to interactively manipulate the importance of input variables and to visualize complex relations between them through interactive and informative graphical interfaces.

These computational techniques are invaluable in analysing content of the input texts by measuring features such as frequencies of certain keywords or expressions in the text and proportions of participants' speaking time in the talk, and visual representations of these findings. They are, however, not designed to facilitate the analysis of the interactional dynamics of talk (but see Biber 2008 for the relationship between corpus analysis and discourse analysis). Furthermore, the visual representations available in these software applications mainly serve to display researchers' annotated concepts and themes.

IDLab is not designed to replace traditional analysis and methodologies; rather it serves to augment existing tools by offering insights into the data otherwise not easily accessible in non-visual text data. The closest tool to this that is currently available is Discursis (Angus *et al.* 2013), a visualization system that shows the temporal structure of a conversation by representing the time series of each speaker's turn as well as shared concepts between them.

However, while Discursis, using word frequency statistics generated by Leximancer, provides turn-by-turn visual information relating to global topic structures, IDLab focuses on patterns and dynamics of turn-taking. Both tools emphasize the importance of visual representation of conversational dynamics.

IDLab is written entirely in R, a free and open-source language designed for statistics (R Core Team 2014). R was chosen for a number of reasons:

1. It is rapidly becoming the *de facto* standard for data analysis. It is widely used by statisticians and has a vibrant community contributing open- source state-of-the-art packages.
2. It provides a flexible graphics package (ggplot2).
3. It provides a package that creates an interactive webpage from R alone, bypassing the need for expertise in web server technology (shiny).

This tool is an example of what OSS enables. It is entirely built on OSS, from R and its packages to Javascript libraries. This project and many others could not have succeeded without the thousands of developers who decided to share their work. It is now open source, its code published on Github (https://github.com/aktionsart/interactionalDiscourseLab), a platform that facilitates collaboration on software development. Interested users can either submit patches or report bugs. Non-technical users can request features or give feedback on Gitub or through a web form on the dedicated website (http://interactionaldiscourselab.net/).

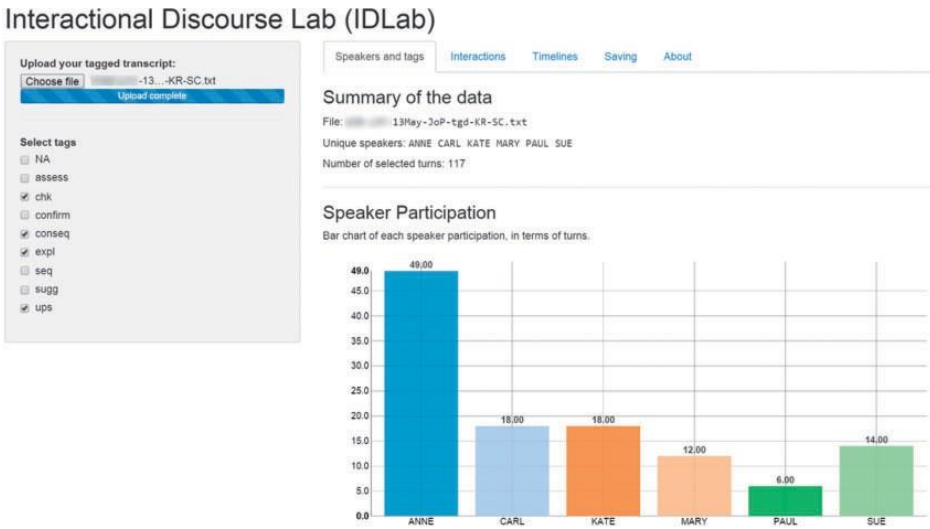

Figure 1: Sample page from IDLab

The visual representation produced gives a synaptic view of the dynamics of spoken engagement, highlighting frequent exchanges and important contributors, while the frequencies themselves can be read in the associated table. Each frequency comes with a confidence interval to convey the uncertainty attached to the measurement (May *et al.* 2000).

One of the strengths of this tool is its ease of use for those with no specialist knowledge. Once a user has tagged a text or collection of texts (a simple matter of adding a code within slashes at the end of the turn; e.g. 'Was that on Saturday? /check/'), the program will generate a page with a panel containing the full list of tags. The user then selects as many of these as required in order to explore relationships between them. In addition to the tagged items, the program will also automatically generate visual representations of turn distribution, speaker interaction networks, and so on Figure 1 illustrates a typical page.

The engine of IDLab has been designed to facilitate growth and experimentation. It is modular both internally and externally. Internally, the processed data are available as input to any new functions, bypassing unnecessary duplication of the workflow (from raw data, to turn selection to simple statistics). Externally, the interface is made up of thematically distinct panels. An interested and technically able user can thus contribute a new, separate module without having to delve into the intricacies of the original version of the software. IDLab is publicly hosted on Github, with an accompanying issue tracker. This enables non-technical users to report bugs or request features using a simple form. They can also leave comments on the project's page. While it is not possible to demonstrate the full range of the tool's functionality in this

article, the next section offers a snapshot of two features that were part of a wider project.

'So' in interdisciplinary research meetings

The examples presented in this article are drawn from a larger project that I am conducting, with over 400 hours of audio-recorded interdisciplinary scientific research project meetings varying in length from one to eight hours and involving researchers from physics, medicine, mathematics, statistics, biology and bioinformatics. The aim of this larger project is to understand better the nature of interdisciplinary research engagement with a view to improving the effectiveness of the interaction involved and the aspect discussed here focused on 'so-prefaced' turns in biology-focused interdisciplinary research projects comprising over 24 hours of transcribed talk amounting to over 330,000 words. A distinction is drawn in biology between 'wets', who conduct experiments, and 'dries', who are broadly speaking theoreticians. The research revealed an asymmetry of epistemic rights that privileged the contributions of the former involved at the expense of those from the latter.

How speakers begin their turns can be indicative of positions they take up and the turn initial position has been recognized as particularly important (Schegloff 1987; Heritage 2013). This tool was used to explore 'so'-initial turns, which occurred with striking frequency: a count of turn-initial occur-rences revealed that 6.8% of turns in the full data set began with 'so', repre- senting approximately one occurrence every 15 turns.

Mapping interaction

In order to map aspects of the interaction that contribute to its topography, it is first necessary to tag the data. The limitations of a priori categorization have been well debated (Van Rees 1992; Schegloff 2005), so it is important to emphasize that the aim at this point is not to develop a definitive analysis but to generate an indicative map of the relevant interactional terrain. The IDLab tool places no limit on the number of categories used, but a preliminary analysis of the interdisciplinary corpus identified four functions that accounted for all but a handful of turn-initial 'so' occurrences: *Check* (where the aim is to understand, clarify, etc.), *Explanation* (comprising anything that serves to explain, account for, provide reasons or motives for, etc.), *Consequence* (expressing the causal relationship between the stated outcome, in terms of actions or states, and prior actions or conditions, both of which are known to the speaker), and *Upshot* (expressing broader consequences, results or implications of prior talk, involving a summary or interpretation of some aspect of that talk and addressing the question, 'What does that amount to?'). What emerged most strikingly from this fairly basic mapping of a single feature was the extent to which it reflects features of relevant activities, orientations, and

relationships, but the following analysis focuses on just two visual outputs: interaction networks and timeline.

The importance of visual data display at any stage of research is widely acknowledged and its goal is to provide ready access to information arising from, and facilitate discoveries or particular perspectives on, a specific area of investigation (Dey 1993; Burke *et al.* 2005; Lengler and Eppler 2007; Slone 2009), though as Verdinelli and Scagnoli (2013) note, visual display in qualitative research is underutilized and underdeveloped. Given the increasing sophistication of visual literacy and the proliferation of digital publications encouraging researchers to embrace more visual forms of communication, however, conditions are ideal for developing this aspect of data presentation. Visual displays generated by the IDLab provide ready access to information about the distribution of the turn-initial 'so' and the interactional relationships between participants which may not be immediately visible to analysts in data sets of this size. These representations (available in colour via [Link: *Applied Linguistics* webpage]) also enable researchers to see the connection between different segments of relevant data and allow them to acquire insights, develop, and elaborate understanding.

Network

Interaction networks need to be read in conjunction with other outputs but is provided here for illustrative purposes. The size of the circles in Figures 2 and 3 represents the extent of turn-initial 'so' use but not overall participation, and while the thickness of the lines reflects the turn relationship as a percentage of overall use for the individual from which the arrow derives, this has no implications for the number of turns involved. Since this output shows only *consecutive* (i.e. immediately adjacent) turns, it represents only a tiny subset of the sequences shown on the timeline. It is therefore no more than indicative, but it can nevertheless direct attention to potentially interesting relationships.

While there is much that could be said about these diagrams, I focus here only on one aspect: leadership. Although there is technically no leader in the meetings, Carl is by far the most senior figure and Kate is the principle investigator, so these might be expected to provide research leadership. A comparison of Figures 2 and 3 suggests an interesting contrast between meetings where Carl is present and those where he is not. It is immediately apparent, for example, that the talk is much more distributed in Figure 2, where he is absent, than it is in Figure 3, suggesting that his presence shifts the interactional dynamic of the group. The only two people between whom there is no connection in Figure 2 are Sue and Paul, though this is to be expected because they are attending the meeting as experts in different disciplines, so one would naturally expect them to interact with the experimental team rather than with one another. Sue's more distributed involvement in Figure 2 can be explained by the fact that in Carl's absence she is the only person with

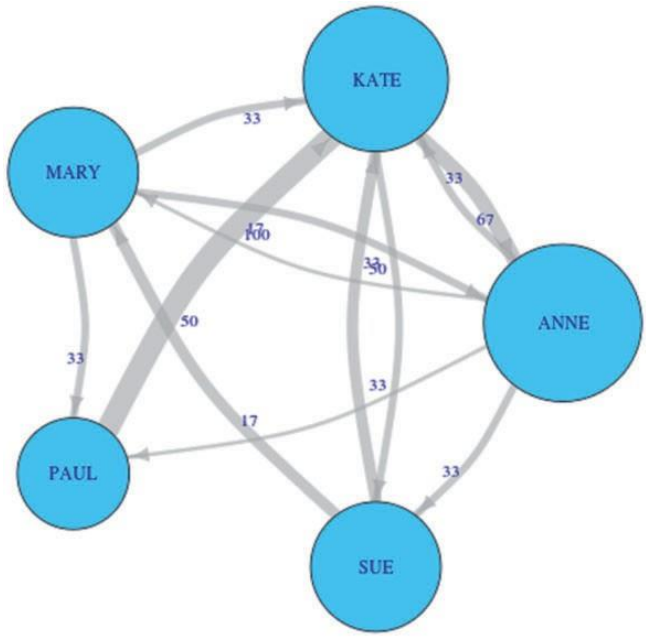

Figure 2: Speakers contiguous use of 'so'-20 Mar

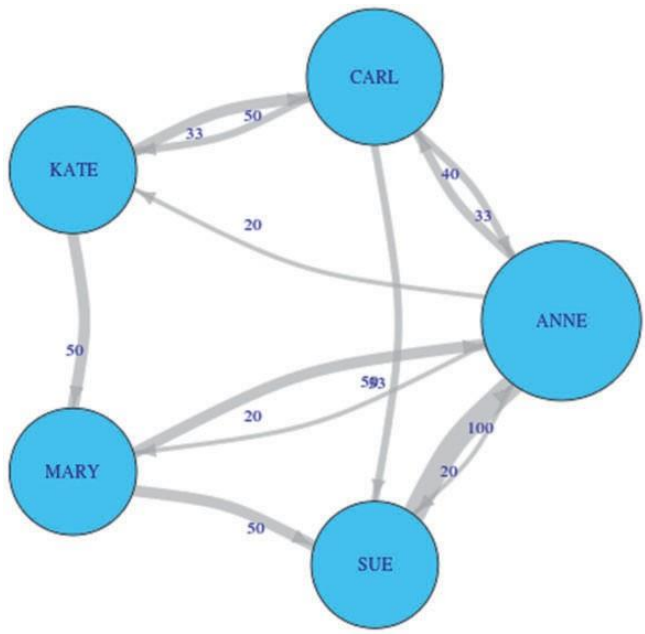

Figure 3: Speakers contiguous use of 'so'-13 May

expert statistical knowledge present. In the meeting he attends (Figure 3), Carl's contribution is restricted to interactions with Kate (the leading 'wet') and Anne (the leading 'dry'). Although Paul is present at that meeting and speaks, he has no consecutive 'so'-initial turns and therefore does not appear on the figure.

As indicated above, this is merely indicative, but it does point to potentially interesting relationships in the talk in terms of how leadership gets done. These can then be followed up with close analysis of the talk itself. This is further facilitated if the visual representation of the development of the talk is also considered.

Timeline

A timeline of one meeting, representing all 'so'-initial turns (the non-contiguous ones separated by a black line) by function and speaker (y-axis, Figure 4) reveals a number of aspects that add important details to the map of the talk. For the purposes of analysis, I focus only on upshots, which are more common towards the end than at the beginning of the meeting, though there is a small cluster at the end of an opening 'wet' presentation.

The 'ownership' of upshots reveals a distributed leadership dynamic, even though Carl is present. Distributed leadership describes those constellations in which teams lead their work collectively and independently of formal leaders (Vine *et al.* 2008). Thus, rather than relying on an officially assigned leader or chair to lead decision making, agenda setting, and so on, in distributed leadership, the various activities typically associated with leadership are conjointly performed among team members who may be on the same or different hierarchical level within their organization (see also Gronn 2002; Day *et al.* 2004; Nielsen 2004). In this project meeting, upshots express interpretations and implications of the topic of discussion that will determine making important decisions on what next steps are for the project. Thus it is unsurprising that the final clusters of upshots are led by both Kate and Carl exercising their leadership roles, and this is indeed what the timeline shows. However, potentially more interesting is the fact that Anne, a post-doc, leads with upshots throughout the meeting, suggesting that leadership of this team may in fact be

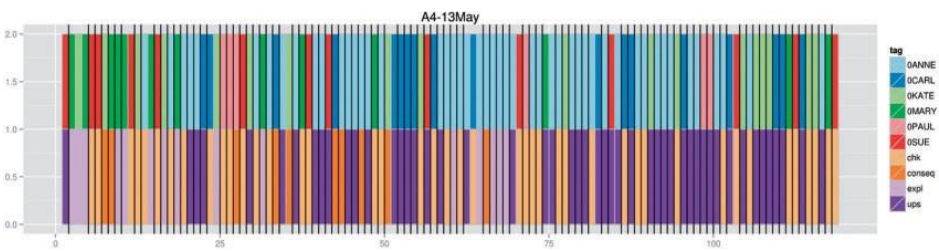

Figure 4: Timeline – 13 May

distributed, something that close analysis of the interaction actually bears out (Choi and Schnurr 2014; Choi and Richards in press).

The timeline also illustrates how OSS tool development can respond rapidly to user needs. While Figure 4 provides an overview of the sort described above, it is less amenable to more detailed inspection and this proved frustrating for some early users. The most recent version of the tool, developed in a few days (as evidenced by the change history of the tool available at Github: https://github.com/aktionsart/interactionalDiscourseLab/commits/master) in order to address the problem, includes an 'expansion' tool accompanied by relevant descriptors that makes possible finer grained analysis.

This illustration indicates that the tool, even at its current stage of development, offers useful insights into the nature of group interaction, but far more important in terms of the case for OSS as a platform for methodological innovation is how this might be developed. At the most basic level, the current list (speakers and tags, interactions, and timeline) might be extended, but the tool might easily be adapted to other uses. It could, for example, be adapted for the analysis of different repair trajectories in talk, possibly with types of turn-constructional units as an additional dimension, thus shifting the focus towards features that are traditionally associated with conversation analysis. A more interesting though challenging development might use Bayesian ana- lysis to predict the likely occurrence of a particular feature in the talk on the basis of the distribution of key features and/or lexical items, thus extending analysis beyond the turn to the sequence.

CONCLUSION

Although visual outputs of the IDLab are generated by embedded statistical tools that allow quantitative representation, the visuals themselves afford—in fact, encourage—interpretive readings that might lead to either selection and close qualitative analysis of relevant extracts or further quantitative analysis. In this respect, the tool might be seen as more paradigmatically neutral than most commercial software that is currently available. The outputs of the tool itself might feature in subsequent publications of findings (Choi and Richards 2014) or might be used simply for the purpose of identifying patterns in the data that might otherwise not be obvious and/or selecting relevant extracts for consideration, without featuring as part of the publication itself (Choi and Schnurr 2014; Choi and Richards in press).

Tools such as IDLab offer an opportunity to map patterns of talk within specific domains and are particularly valuable where the focus is on groups that have worked together over time and where distinctive interactional con-tours have developed (Richards 2006, 2010). There are many different land- scapes of interaction, some features of which change over time while others endure, and tools are now available to describe such terrains in sufficient detail to be analytically productive, allowing the development of interactional

mapping systems. Such tools therefore offer the prospect of developing what might be termed *interactional topographies.*

If collections of such topographies are to be a realistic prospect, the case for OSS needs to be given serious consideration. The advantages of OSS, summed up in the acronym VARIES, make it an ideal platform for collaborative methodological innovation and the moment is right for AL to give a lead in the social sciences in taking this up. The rapid expansion of OSS outside the social sciences means that in doing so it would be catching a wave that is already gathering momentum, with a model for collaborative development already established in the physical sciences. Unlike CAQDAS programs, which are stretched to cope with the 'big data' that is now becoming available, the developmental potential of OSS means that it is well positioned to respond to the challenges of this. This development also benefits from the fact that computing speed and capacity are not only so much greater than in the past but also more generally available, offering the prospect of more ambitious projects and contributing to the expansion of innovative mixed methods approaches (see Riazi, this issue).

Recent developments in AL are encouraging for researchers willing to explore what OSS has to offer. There is evidence, for example, of growing interest in OSS in text analysis (e.g. the large number of R packages listed on the natural language processing task view: http://cran.r-project.org/web/views/NaturalLanguageProcessing.html) and of new developments in interactive visualization tools for research (Siirtola *et al.* 2011, 2014; Angus *et al.* 2012).

In advancing the case for OSS, this article has adopted a largely practical orientation, but it should be seen as part of a broader paradigm shift within AL away from what Firth and Wagner (1997: 285) called an 'individualistic and mechanistic' view of communication in SLA focusing on *individual* language cognition, towards what Block (2003) described as the 'social turn' in which the emphasis has shifted to language as co-constructed social action in which 'human actors *dynamically adapt to*—that is, flexibly depend on, integrate with, and construct—the ever-changing mind-body-world environments posited by sociocognitive theory' (Atkinson *et al.* 2007: 171). The development of distributed cognition as a scientific discipline (for a definition and overview, see Zhang and Patel 2006) finds a parallel in Cowley's concept of distributed language that eschews 'organism-centred models' (Cowley 2011: 4) and situates language ecologically. What is being proposed in this article is an approach to analysis that has theoretical resonance with this fundamental shift in the way language is perceived. The analytical process is no longer reified and systemically 'fixed' but is seen as an epiphenomenon of distributed multi-agent analytical engagement that is both co-constructed and constantly evolving. In shifting the responsibility of the researcher away from reflexive positioning as an individual and towards informed engagement with emergent systems, this also directs attention to the possibility of responding dynamically to evolving patterns of interaction within specific groups.

The full implications of the foundational shift from treating language as a semiotic system to seeing it as an environmental artifact are yet to be worked through, though methodologically it is reflected in the steady growth in conversation analytic research (Seedhouse 2005; Hellermann 2008) and the development of multi-modal approaches (Seedhouse and Knight, this issue). What this article points to is the possibility of the distributed development of analytical tools and methodological understanding, which may in turn be part of a broader reconceptualization of research. Reflexivity, for example, has become a key concern for qualitative researchers (for an AL perspective, see Roulston 2010) and is currently conceived as a matter of individual cognition, though Byrd-Clark and Dervin's (2014: 234) claim that '[r]eflexivity should not just be something that is taking place inside the individual' may be a straw in the wind. The time may therefore be ripe for exploring and testing the full potential of OSS, trialling approaches that are flexible, de-centralized, and distributed, working towards a reconfiguration of methodological development within a collaborative frame. In so doing, AL would be providing the lead for the social sciences generally.

NOTE

1 See also Text Analysis Portal for Research (TAPOR 2.0: http://www. tapor.ca/) for more software tools.

ACKNOWLEDGEMENTS

The author would like to thank Keith Richards for his unwavering support throughout the write-up of this article. The author would also like to thank John Hellermann and the anonymous reviewers, whose comments have strengthened this article considerably. Finally, the author would also like to thank the scientific research project team members for their generous assistance allowing me to record and observe their project meetings.

REFERENCES

Angus, D., S. Rintel, and J. Wiles. 2013. 'Making sense of big text: A visual-first approach for analysing text data using Leximancer and Discursis,' *International Journal of Social Research Methodology* 16/3: 261–7.

Angus, D., D. Rooney, B. McKenna, and J. Wiles. 2012. 'Visualizing punctuated equilibria in discursive change: Exploring a new text analysis possibility for management research,' *Journal of Business and Management Landscapes* 1/1: 1–16.

Anthony, L. 2013. 'Critical look at software tools in corpus linguistics,' *Linguistic Research* 30/2: 141–61.

Anthony, L., Ch. Kiyomi, and K. Oghigian.2011. 'A novel, web-based, parallel concordan- cer for use in the ESL/EFLclassroom' in J. Newman, (ed.): *Corpus-Based Studies in Language Use, Language Learning, and Language Documentation*. Rodopi, pp. 123–38.

Atkinson, D., E. Churchill, T. Nishino, and H. Okada. 2007. 'Alignment and interaction in a sociocognitive approach in second

language acquisition,' *Modern Language Journal* 91/2: 169–88.

Bales, R. 1950. *Interaction Process Analysis*. Addison-Wesley.

Biber, D. 2008. 'Corpus-based analyses of discourse: Dimensions of variation in conversation' in V. K. Bhatia, J. Flowerdew, and R. H. Jones (eds): *Advances in Discourse Analysis*. Routledge, pp. 100–14.

Biber, D., S. Conrad, and R. Reppen. 1998. *Corpus Linguistics*. Cambridge University Press.

Block, D. 2003. *The Social Turn in Second Language Acquisition*. Edinburgh University Press.

Burke, J., P. O'Campo, G. Peak, A. Gielen, K. McDonnell, and W. Trochim. 2005. 'An introduction to concept mapping as a participatory public health research method,' *Qualitative Health Research* 15/10: 1392–410.

Byrd-Clark, J. and F. Dervin. 2014. *Reflexicity in Language and Intercultural Education: Rethinking Multilingualism and Interculturality*. Routledge.

Choi, S. and K. Richards. 2014. 'Computational analysis of turn-taking patterns in interdisciplinary research meetings' in S. Ruhi, M. Haugh, T. Schmidt, and K. Wo ̈rner (eds): *Best Practices for Spoken Corpora in Linguistic Research*. Cambridge Scholars Publishing, Newcastle upon Tyne, pp. 95–116.

Choi, S. and K. Richards. In press. 'The dy- namics of identity struggles in interdisciplinary meetings in higher education' in S. Schnurr and D. Van De Mieroop (eds): *Identity Struggles*. John Benjamins Publishing.

Choi, S. and S. Schnurr. 2014. 'Exploring distributed leadership: Solving disagreements and negotiating consensus in a 'leaderless' team,' *Discourse Studies* 16/1: 3–24.

Cowley, S. 2007. 'Distributed language: Biomechanics, functions, and the origins of talk' in C. Lyon, C. L. Nehaniv, and A. Cangelosi (eds): *Emergence of Communication and Language*. Springer, pp. 105–27.

Cowley, S. 2011. *Distributed Language*. John Benjamins Publishing.

Day, D., P. Gronn, and E. Salas. 2004. 'Leadership capacity in teams,' *Leadership Quarterly* 15/6: 857–80.

Dey, I. 1993. *Qualitative Data Analysis: A User-Friendly Guide for Social Scientists*. Routledge.

Evers, J., K. Mruck, C. Silver, and B. Peeters. 2011. 'The KWALON experiment: Discussions on qualitative data analysis software by developers and users,' *Forum: Qualitative Social Research* 12/1.

Fielding, N. and C. A. Cisneros-Puebla. 2009. 'CAQDAS-GIS Convergence: Toward a new integrated mixed method research practice?,' *Journal of Mixed Methods Research* 3/4: 349–70

Fielding, N. and R. Lee. 2002. 'New patterns in the adoption and use of qualitative software,' *Field Methods* 14/2: 197–216.

Firth, A. and J. Wagner. 1997. 'On discourse, communication, and (some) fundamental concepts in SLA research,' *The Modern Language Journal* 81/3: 285–300.

Garretson, G. 2008. 'Desiderata for linguistic software design,' *International Journal of English Studies* 8/1: 67–94.

Gries, S. T. 2009. *Quantitative Corpus Linguistics with R*. Routledge.

Gronn, P. 2002. 'Distributed leadership as a unit of analysis,' *Leadership Quarterly* 13/4: 423–51.

Hellermann, J. 2008. *Social Actions for Classroom Language Learning*. Multilingual Matters.

Heritage, J. 2013. 'Turn-initial position and some of its occupants,' *Journal of Pragmatics* 57: 331–7.

Hippel, E. 2001. 'Innovation by user communities: Learning from open-source software,' *MIT Sloan Management Review* 42/4: 82.

Joppa, L. N., G. McInerny, R. Harper, L. Salido, K. Takeda, K. O'Hara, D. Gavaghan, and S. Emmott. 2013. 'Troubling trends in scientific software use,' *Science* 340/6134: 814–15.

Lengler, R. and M. Eppler, 2007. 'Towards a periodic table of visualization methods for management' in *IASTED Proceedings of the Conference on Graphics and Visualization in Engineering*, Clearwater, FL, available at www.visual-literacy.org/periodic_table/periodic_table.pdf.

Lerner, J. and J. Tirole. 2002. 'Some simple economics of open source,' *Journal of Industrial Economics* 52/2: 197–234.

Mason, O. 2008. 'Developing software for corpus research,' *International Journal of English Studies* 8/1: 141–56.

May, W., L. Johnson, and D. William. 2000. 'Constructing two-sided simultaneous confidence intervals for multinomial proportions for small counts in a large number of cells,' *Journal of Statistical Software* 5/6: 1–24.

McEnery, T. and A. Hardie. 2011. *Corpus Linguistics: Methods, Theory and Practice*. Cambridge University Press.

Morin, A., J. Urban, P. D. Adams, I. Foster, A. Sali, D. Baker, and P. Sliz. 2012a.

'Shining light into black boxes,' *Science* 336/6078: 159–60. doi:10.1126/science.1218263.

Morin, A., J. Urban, and P. Sliz. 2012b. 'A quick guide to software licensing for the scientist-programmer,' *PLoS Computational Biology* 8/7: e1002598. doi:10.1371/journal.pcbi.1002598.

Nielsen, J. 2004. *The Myth of Leadership: Creating Leaderless Organizations*. Davies-Black.

Payne, C. 2002. On the security of open source software. *Information Systems* 12, 61–78.

Peng, R. D. 2009. 'Reproducible research and Biostatistics,' *Biostatistics* 10/3: 405–8.

Peng, R. D. 2011. 'Reproducible research in computational science,' *Science* 334/6060: 1226–7.

Porte, G. 2012. *Replication Research in Applied Linguistics*. Cambridge University Press.

Prlic´, A. and H. Lapp. 2012. 'The PLOS computational biology software section,' *PLoS Computational Biology* 8/11: e1002799. doi:10.1371/journal.pcbi.1002799.

Prlic´, A. and J. B. Procter. 2012. 'Ten simple rules for the open development of scientific software,' *PLoS Computational Biology* 8/12: e1002802. doi:10.1371/journal.pcbi.1002802.

R Core Team. 2014. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing.

Ram, K. 2013. 'Git can facilitate greater reproducibility and increased transparency in science,' *Source Code for Biology and Medicine* 8/1: 7. doi:10.1186/1751-0473-8-7.

Raymond, E. S. 2001. *The Cathedral & the Bazaar: Musings on Linux and Open Source by an Accidental Revolutionary*. O'Reilly Media.

Richards, K. 2006. *Language and Professional Identity*. Palgrave Macmillan.

Richards, K. 2010. 'Professional orientation in back region humor,' *Text and Talk* 30/2: 221–41.

Roter, D. and S. Larson. 2002. 'The Roter interaction analysis system (RIAS): Utility and flexibility for analyses of medical interactions,' *Patient Education and Counselling* 46: 243–51.

Roulston, K. 2010. *Reflective Interviewing: A Guide to Theory and Practice*. Sage.

Schegloff, E. A. 1987. 'Recycled turn begin- nings: A precise repair mechanism in conver- sation's turn-taking organization' in G. Button and J.R.E. Lee (eds): *Talk and Social Organization*. Multilingual Matters, pp. 70–85.

Schegloff, E. A. 2005. 'Presequences and indirection: Applying speech act theory to ordinary conversation,' *Journal of Pragmatics* 12/1: 55–62.

Seedhouse, P. 2005. *The Interactional Architecture of the Language Classroom: A Conversation Analysis Perspective*. Blackwell.

Siirtola, H., T. Nevalainen, T. Sa¨ily, and K. Ra¨iha¨. 2011. 'Visualisation of text corpora: A case study of the PCEEC,' *Studies in Variation, Contacts and Change in English 7*, available at www.helsinki.fi/varieng/series/volumes/07/siirtola_et_al/

Siirtola, H., T. Nevalainen, T. Sa¨ily, and K. Ra¨iha¨. 2014. 'Text variation explorer: Towards interactive visualization tools for corpus linguistics,' *International Journal of Corpus Linguistics* 19/3:417–29.

Silver, C. and J. Patashnik. 2011. 'Finding fidelity: Advanced audiovisual analysis using software,' *Forum: Qualitative Social Research* 12/1: art 37.

Silver C. and A. Lewins. 2007. *Using Software in Qualitative Research: A Step-by-Step Guide*. Sage.

Silver, C. and A. Lewins. 2014. *Using Software in Qualitative Research: A Step-by-Step Guide*, 2nd edn. Sage.

Slone, D. J. 2009. 'Visualizing qualitative information,' *The Qualitative Report* 14/3: 489–97.

Sojer, M. and J. Henkel. 2010. 'Code reuse in open source software development: Quantitative evidence, drivers, and impediments,' *Journal of the Association for Information Systems* 11/12: 868–901.

Sornette, D., T. Maillart, and G. Ghezzi. 2014.'How much is the whole really more than the sum of its parts? 1 + 1 = 2.5: Superlinear product- ivity in collective group actions,' *PLoS ONE* 9/8: e103023. doi:10.1371/journal.pone.0103023.

Van Assen, M. A. L. M., R. C. M. van Aert, M. B. Nuijten, and J. M. Wicherts. 2014.'Why publishing everything is more effective than selective publishing of statistically signifi- cant results,' *PLoS ONE* 9/1: e84896. doi:10.1371/journal.pone.0084896.

Van Rees, M. A. 1992. 'The adequacy of speech act theory for explaining conversational phenomena: A response to some conversation analytical critics,' *Journal of Pragmatics* 17/1: 31–47.

Verdinelli, S. and N. Scagnoli. 2013. 'Data display in qualitative research,' *International Journal of Qualitative Methods* 12: 359–81.

Vine, B., J. Holmes, M. Meredith, D. Pfeifer, and B. Jackson. 2008. 'Exploring co-leadership

talk through interactional sociolinguistics,' *Leadership* 4/3: 339–60.
Weisser, M. 2009. *Essential Programming for Linguistics*. Edinburgh University Press.
Wilson G., D. A. Aruliah, C. T. Brown, N. P. Chue Hong, M. Davis, R. T. Guy, S.H.D. Haddock, K. D. Huff, I. M. Mitchell, M. D. Plumbley, B. Waugh, E. P. White, and P. Wilson. 2014. 'Best practices for scientific computing,' *PLoS Biol* 12/1: e1001745. doi:10.1371/journal.pbio.1001745.
Wren, J. D. 2008. 'URL decay in MEDLINE – A 4-Year follow-up study,' *Bioinformatics* 24/11: 1381–5.
Zhang, J. and V. L. Patel. 2006. 'Distributed cognition, representation, and affordance,' *Pragmatics and Cognition* 14/2: 333–4.